Desarrollo Seguro en Python

Lección 1: Introducción al Desarrollo de Software Seguro

eip ESCUELA INTERNACIONAL POSGRADOS

ÍNDICE

Introducción al Desarrollo de software seguro

Objetivos

- En esta lección aprenderás a:
- Saber en qué consiste el desarrollo seguro de software
- Conocer las propiedades del software seguro
- Conocer los principios del diseño seguro
- Incluir el desarrollo seguro en tus proyectos

1. Definición desarrollo seguro software

Podemos definir un programa seguro como aquel que es capaz de seguir realizando las funciones para las que ha sido diseñado en todo momento, y capaz de evitar que la existencia de errores en él pueda ser utilizada como puerta de entrada para la realización de acciones que pongan en peligro la integridad, confidencialidad o disponibilidad del resto de elementos del sistema en el que se está ejecutando.

La programación segura engloba un conjunto de técnicas, normas y conocimientos que permiten crear programas que no puedan ser modificados de forma ilegítima con fines maliciosos y que estén carentes de fallos que puedan comprometer la seguridad del resto de elementos del sistema con el que interactúan.

Los objetivos de la utilización de prácticas de software seguras son las siguientes:

- Los fallos explotables y otros puntos débiles se eliminan en la mayor medida posible.
- La probabilidad de que los desarrolladores puedan producir errores y vulnerabilidades explotables o puertas traseras en el software se reduce o elimina en gran medida.
- El software es resistente y tolerante a posibles ataques para apoyar el cumplimiento de la misión de la organización.

1.1. Propiedades del software seguro

En el desarrollo de software usamos metodologías clásicas (cascada, prototipado, espiral, incremental y RAD) y las metodologías ágiles (SCRUM, CRYSTAL, KANBAN, FDD, ASD o LSD entre otras) todas como tal persiguen el desarrollo óptimo del software con unos niveles de confiabilidad y control de la solución informática.

En estas metodologías la seguridad informática y sus principios de diseño seguro no suelen ser parte formal u obligatoria.

Cuando una aplicación transmite o almacena información confidencial, la aplicación es responsable de garantizar que los datos almacenados y transferidos estén cifrados y no puedan obtenerse, alterarse o divulgarse fácilmente de forma ilícita.

El objetivo fundamental de la seguridad informática se basa en preservar los siguientes puntos.

- **Confidencialidad:** el software debe asegurar que cualquiera de sus características, los activos que administra y su contenido son accesibles solo para las entidades autorizadas e inaccesibles para el resto. El acceso a la información está limitado a usuarios autorizados. Los datos deben protegerse de la observación o divulgación no autorizadas tanto en tránsito como almacenadas.
- **Integridad:** El software y los activos del sistema solo pueden ser modificados por usuarios autorizados. Esta propiedad se debe preservar durante el desarrollo de software y su ejecución. Los datos han de protegerse al ser creados, alterados o borrados maliciosamente por atacantes no autorizados.
- **Disponibilidad:** El software debe estar operativo y ser accesible a sus usuarios autorizados siempre que se requiera. De modo que los usuarios puedan realizar sus tareas de formar correcta y dar cumplimiento a los objetivos de la organización que lo utiliza.

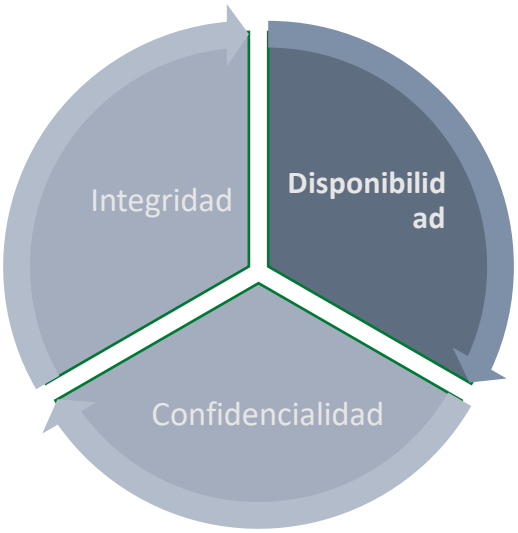


Figura 1.1 Propiedades del software seguro.

Para las entidades que actúan como usuarios se requieren dos propiedades adicionales.

- **Trazabilidad:** todas las acciones relevantes relacionadas con la seguridad de una entidad que actúa como usuario se deben registrar y trazar con el objetivo de disponer de datos para la realización de auditorías; de esta forma, la trazabilidad debe ser posible tanto durante la ocurrencia de las acciones registradas como a posteriori.

- **No repudio.** Constituye la habilidad de prevenir que una entidad que actúa como usuario desmienta o niegue la responsabilidad de acciones que han sido ejecutadas.

Estas son las propiedades básicas más utilizadas para describir la seguridad de los sistemas y aplicaciones.

Algunos ejemplos de ataques y las propiedades que vulneran:

Ataque	Propiedades vulneradas
Inyección de SQL	Confidencialidad
Cross-site scripting (XSS)	Integridad, Disponibilidad
Desbordamiento de búfer que inyecta código con el objetivo de obtener y modificar la información de usuarios.	Confidencialidad, Integridad, Disponibilidad, Trazabilidad y No repudio.

2. PRINCIPIOS DEL DISEÑO SEGURO

Partiendo de las propiedades básicas anteriores veremos una serie de principios orientados al diseño seguro de aplicaciones:

2.1 Minimizar el área de la superficie de ataque

Minimizar el área de la superficie de ataque. Cada característica que se añade a una aplicación incrementa su complejidad y aumenta también el riesgo de aplicación en conjunto. Una nueva característica implica un nuevo punto de ataque.

Uno de los factores clave para reducir el riesgo de una aplicación recae en la reducción de la superficie de ataque, pueden eliminarse posibles puntos de ataque si se deshabilitan módulos o componentes innecesarios para la aplicación.

Por ejemplo

Si la aplicación no utiliza el almacenamiento en caché de resultados, sería recomendable deshabilitar dicho módulo. De esta manera, si se detecta una vulnerabilidad de seguridad en ese módulo, la aplicación no se verá amenazada.

2.2 Seguridad por defecto

Seguridad por defecto. La **seguridad debe ir implícita en el propio desarrollo**. Tanto en la experiencia del cliente como en las prácticas habituales de los desarrolladores. No usar opciones de configuración de seguridad reducidas para evitar que dichas configuraciones compliquen el desarrollo. **Si el propio desarrollo de la aplicación obliga a cambiar opciones de seguridad se debe realizar una auditoría de seguridad**.

Por ejemplo

Si para que la aplicación funcione tenemos que tener una carpeta compartida en el servidor sin permisos de acceso "Para Todos" habría que realizar una auditoria de seguridad y analizar ese requisito.

2.3 Privilegios mínimos

Privilegios mínimos. Según el principio del mínimo privilegio, se recomienda que las cuentas de usuario tengan la mínima cantidad de privilegios necesarios. Asimismo, se aconseja que los procesos se ejecuten únicamente con los privilegios necesarios para completar sus tareas. De esta manera, se limitan los posibles daños que podrían producirse si se ve comprometido el proceso.

Si un atacante llegase a tomar el control de un proceso en un servidor o comprometiese una cuenta de usuario, los privilegios concedidos determinarán, en gran medida, los tipos de operaciones que podrá llegar a realizar dicho atacante.

Importante

Privilegios mínimos para usuario los procesos con los privilegios mínimos necesarios para completar la tarea.

2.4 Validación de datos de entrada

Validación de datos de entrada. Se ha de garantizar que la aplicación sea robusta ante todas las posibles formas de entrada de datos, ya sean proporcionados por el usuario, por la infraestructura, por entidades externas o por bases de datos.

Una premisa fundamental reside en no confiar en los datos que el usuario pueda introducir, ya que este tiene todas las posibilidades de manipularlos.

La debilidad de seguridad más común en aplicaciones es la falta de validación apropiada de las entradas del usuario o del entorno. Esta debilidad origina casi todas las principales vulnerabilidades en las aplicaciones.

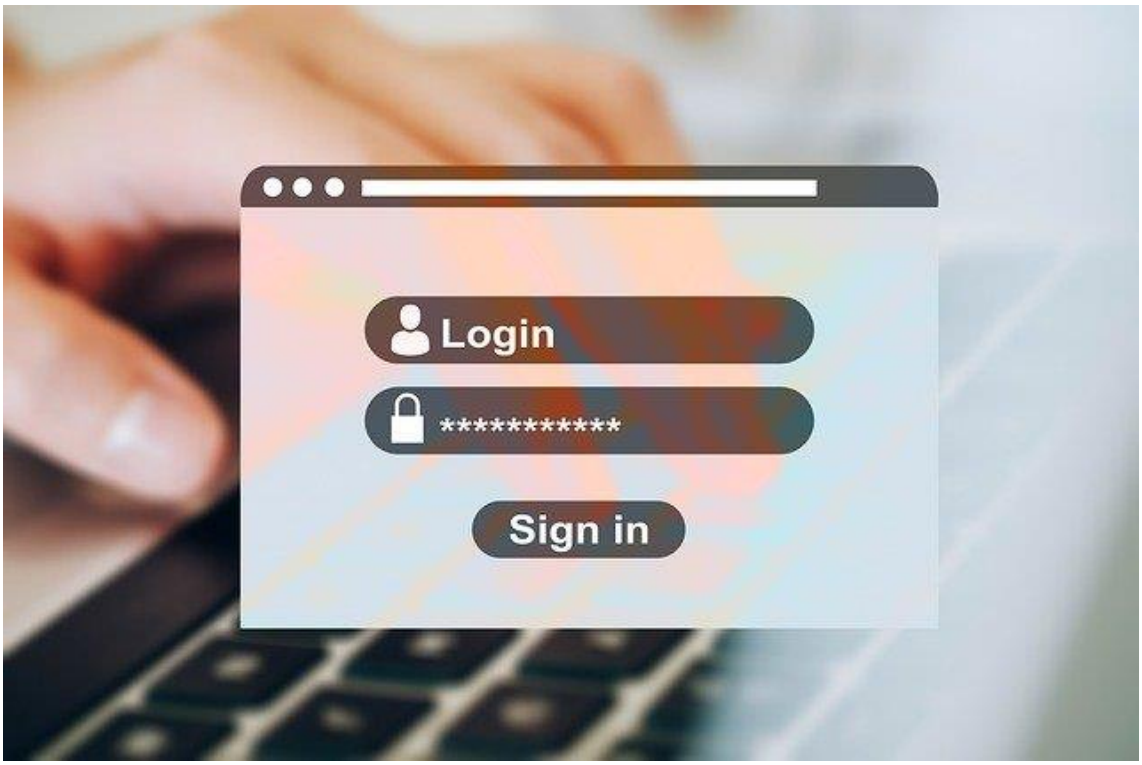


Figura 2.1. Usuario autenticándose introduciendo datos a una aplicación.

2.5 Defensa en profundidad

Defensa en profundidad. Consiste en definir una estrategia de seguridad estándar en la que se establezcan varios controles de defensa en cada una de las capas y los subsistemas de la aplicación. Estos puntos de control ayudan a garantizar que solo los usuarios autenticados y autorizados puedan obtener el acceso a la siguiente capa y sus datos.

Utilizar Framework que nos ayuden a definir la seguridad durante todo el proceso de desarrollo es una buena estrategia para mantener la defensa en profundidad de nuestra aplicación. A continuación se muestra la figura de los procesos del SDL Seguro de Microsoft que llevan aplicando desde 2004.

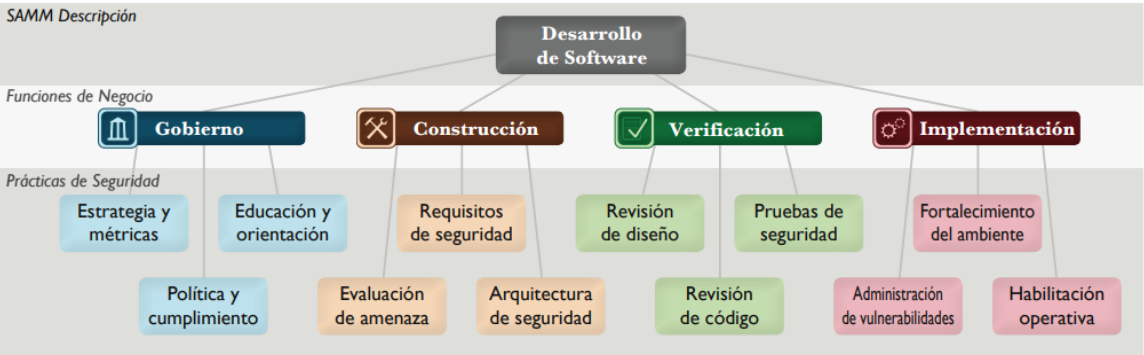


Figura 2.2. Software Assurance Maturity Model. OWASP OPEN SAMM

2.6 Control seguro de errores

Es necesario controlar las respuestas cuándo se produce algún error y no mostrar en ellas información que pudiera ayudar al atacante a descubrir datos acerca de cómo ha sido desarrollada la aplicación o cómo funcionan sus procedimientos.

La información detallada de los errores producidos no debería ser mostrada al usuario, sino que tendría que ser enviada al fichero de log correspondiente.

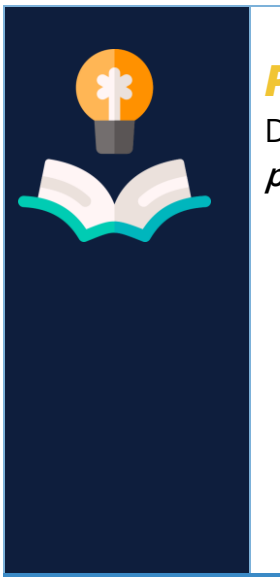
Una simple página de error 403 indicando un acceso denegado puede decir a un escáner de vulnerabilidades que un directorio existe y puede proporcionar a una atacante información que le permita realizar un mapa de la estructura de directorios de la aplicación.

Presta atención

N*unca mostrar al usuario información específica del error para ello utiliza ficheros de log de la aplicación.*

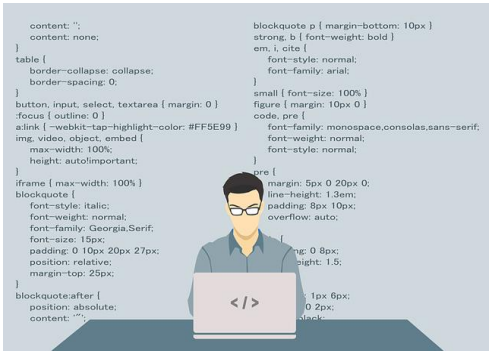
2.7 Separación de funciones

Un punto importante consiste en la separación de funciones entre los distintos perfiles de la aplicación. Además de aplicar privilegios mínimos sobre los distintos roles de los usuarios que tenga nuestra aplicación, hemos de tener bien diferenciadas las funciones que puede hacer cada usuario y evitar que un usuario de bajo nivel puede tener acceso a funciones de un administrador o usuario de nivel superior.



Presta atención

D*iferencia bien los roles para evitar que usuarios de bajo nivel puedan realizar acciones de usuarios superiores*.



2.8 Evitar la seguridad por oscuridad

Evitar la seguridad por oscuridad. La seguridad de una aplicación no debería depender del secreto o confidencialidad de su diseño o implementación. Si se intenta ocultar secretos mediante el uso de nombres de variables engañosos o de ubicaciones de archivos no habituales, no se estará mejorando la seguridad.

La seguridad basada en oscuridad es un control débil, especialmente si se trata del único control. Esto no significa que mantener secretos constituya una mala idea; significa que la seguridad de los sistemas clave no debería basarse, exclusivamente, en mantener detalles ocultos.

La seguridad de una aplicación no debería basarse en mantener en secreto el conocimiento del código fuente, ha de fundamentarse en muchos otros factores, incluyendo políticas de contraseñas, diseño de una arquitectura sólida tanto a nivel de aplicación como a nivel de comunicaciones de red y realización periódica de controles de auditoría.

Por ejemplo

Un ejemplo práctico es Linux, cuyo código fuente es open source y está disponible para todo el mundo y, aún así, se trata de un sistema operativo seguro y robusto.

3. PUNTOS CLAVE

- La seguridad no es una capa ni un proceso más de la aplicación debe ser parte y estar integrada en el desarrollo desde el principio.
- Es más costoso no aplicar seguridad en desarrollo software que descubrir una vulnerabilidad o brecha de seguridad en producción.
- La seguridad informática se basa en preservar los siguientes puntos: confidencialidad, integridad, disponibilidad, trazabilidad y no repudio.